IN THE CLAIMS

Please amend the claims as follows.

1. (Currently Amended) A computer-implemented method, comprising:

analyzing source code to determine a program slice;

creating a program slice diagram that provides a graphical representation of the program slice;

displaying the program slice diagram;

displaying a code browser operable to display a subset of the source code;

displaying a ~~BLAST (~~Block Level Abstract Syntax Tree (BLAST) viewer having one or more control blocks;

determining a cross-reference between the program slice diagram, a control block in the one or more control blocks, and a portion of subset of the source code; node of the one or more nodes, and

indicating the cross reference in the code browser, the BLAST viewer and the program slice diagram.

2. (Previously Presented) The method of claim 1, wherein the program slice diagram further comprises a directed graph comprising a plurality of nodes and arcs, wherein the arcs represent data flow dependencies between the nodes.

3. (Canceled)

4. (Original) The method of claim 2, wherein the nodes represent source code statements within a selected subroutine.

5. (Original) The method of claim 2, wherein the nodes represent variable references outside of a selected subroutine.

6-10. (Canceled)

11. (Previously Presented) A computer-implemented method, comprising:

displaying a template viewer, said template viewer operable to receive semantic information;

performing semantic abstraction to group a subset of nodes together based on the semantic information

analyzing source code to determine a program slice using the subset of nodes;

creating a program slice diagram that provides a graphical representation of the program slice; and

displaying the program slice diagram.

12. (Original) The method of claim 11, further comprising:

identifying a logical category of computations; and

displaying the logical category of computations with a cross-reference to a display of the source code.

13. (Original) The method of claim 11, wherein performing semantic abstraction further comprises:

identifying a logical category of data elements; and

displaying the logical category of data elements with a cross-reference to a display of data.

14-18. (Canceled)

19. (Original) The method of claim 2, further comprising:

simplifying the program slice diagram by retaining only those nodes that correspond to variable references outside of a selected subroutine.

20. (Previously Presented) The method of claim 11, wherein the semantic information comprises a logical event, and wherein performing event abstraction includes collapsing together nodes that correspond to the logical event.

21. (Canceled)

22. (Currently Amended) A ~~signal-bearing~~ tangible computer-readable media comprising computer-readable instructions, wherein the instructions when read and executed by a computer comprise:

analyzing source code to determine a program slice;

creating a program slice diagram that provides a graphical representation of the program slice;

displaying the program slice diagram;

displaying a code browser operable to display a subset of the source code;

displaying a ~~BLAST (~~Block Level Abstract Syntax Tree (BLAST) viewer having one or more control blocks;

determining a cross-reference between the program slice diagram, a control block in the one or more control blocks, and a portion of subset of the source code; node of the one or more nodes, and

indicating the cross reference in the code browser, the BLAST viewer and the program slice diagram.

23. (Currently Amended) The ~~signal-bearing~~ tangible computer-readable media of claim 22, wherein the program slice diagram further comprises a directed graph comprising a plurality of nodes and arcs wherein the arcs represent data flow dependencies between the nodes.

24. (Canceled)

25. (Currently Amended) The ~~signal-bearing~~ tangible computer-readable media of claim 23, wherein the nodes represent source code statements within a selected subroutine.

26. (Currently Amended) The signal-~~signal-bearing~~ tangible computer-readable media of claim 23, wherein the nodes represent variable references outside of a selected subroutine.

27-31. (Canceled)

32. (Currently Amended) A ~~signal-bearing~~ tangible computer-readable media comprising computer-readable instructions, wherein the instructions when read and executed by a computer comprise:

displaying a template viewer, said template viewer operable to receive semantic information;

performing semantic abstraction to group a subset of nodes together based on the semantic information

analyzing source code to determine a program slice using the subset of nodes;

creating a program slice diagram that provides a graphical representation of the program slice; and

displaying the program slice diagram:

33. (Currently Amended) The ~~signal-bearing~~ tangible computer-readable media of claim 32, further comprising:

identifying a logical category of computations; and

displaying the logical category of computations with a cross-reference to a display of the source code.

34. (Currently Amended) The ~~signal-bearing~~ tangible computer-readable media of claim 32, wherein performing semantic abstraction further comprises:

identifying a logical category of data elements; and

displaying the logical category of data elements with a cross-reference to a display of data.

35-39. (Canceled)

40. (Currently Amended) The ~~signal-bearing~~ tangible computer-readable media of claim 22, further comprising:

simplifying the program slice diagram by retaining only those nodes that correspond to variable references that are outside of a selected subroutine.

41. (Currently Amended) The ~~signal-bearing~~ tangible computer-readable media of claim 32, wherein the semantic information comprises a logical event, and wherein performing event abstraction includes collapsing together nodes that correspond to a logical event.

42-63 (Canceled).

64. (Previously Presented) A software visualization environment for visualizing source code, comprising:

a code browser to display source code;

a block-level abstract syntax tree viewer;

a program slice browser to display a program slice as a directed graph comprising a plurality of nodes;

a template viewer; and

a controller that cross-references information between the code browser, the block-level abstract syntax tree viewer, the program slice browser, and the template viewer.

65-67. (Canceled)

68. (Previously Presented) The software visualization environment of 64, wherein the code browser further:

displays line numbers that cross reference to other visualization components.

69-77. (Canceled)

78. (Previously Presented) The software visualization environment of claim 64, wherein the program slice browser further:

displays the directed graph in upside-down-tree layout, wherein the nodes are positioned according to a data-flow pattern.

79-90. (Canceled)

91. (Previously Presented) The software visualization environment of claim 64, wherein the template viewer further:

displays a binding between a logical view of the source code and the source code.

92. (Original) The software visualization environment of claim 91, wherein the binding comprises a template, comprising:

abstract data structures; and

logical steps that manipulate the abstract data structures.

93-94. (Canceled)